REMARKS

Claims 1, 2, 5-7, 9, 12-15, and 18-25 are pending in the present Application.

Examiner has objected to claims 2, 5-7, 9, and 12-14 due to specified informalities. Applicants have amended these claims to attend to the informalities.

Examiner has rejected claims 1, 2, 9, 12, 13, 15, 18, 23, and 25 under 35 U.S.C.§103(a) as being unpatentable over U.S. Patent No. 6,470,329 to Livschitz ("Livschitz") in view of U.S. Patent Application Publication No. 2002/0029214 by Yianilos et al. ("Yianilos").

Livschitz is directed to the synchronization of a data set and a remote copy of the data set. Livschitz teaches that physical data are fixed-length records in data files that can be decomposed into elementary data blocks that permit no further decomposition. See col. 5, lines 55-60. Livschitz teaches that a hash function may operate upon the data sets and the subsets of data sets, and that this hash function is available in two address spaces (e.g., local and remote copies of a data set). See col. 5, lines 63-65 and Figs. 1 and 8. Livschitz describes at least three processes of synchronization. First, a hash function, H, operates upon separated data sets, A and B, to create signatures (e.g., hashes) h(A) and h(B). These hash signatures are compared and if there is a mismatch, the data sets are each subdivided and again hashed with H followed by the subdivided data set hashes being compared. This process is repeated recursively until the portions of all of the data sets needing synchronization are identified. See col. 5, line 66 - col. 6, line 12. Livschitz discloses a second synchronization process in which the hash signature for each elemental data block is created and stored in an array of h(A) and h(B) signatures, respectively. Each hash signature h(A) is communicated and compared to hash signature h(B) to

determine which elemental block requires synchronization. See col. 7, lines 31-48. Livschitz acknowledges that this process may consume a large amount of bandwidth. See col. 7, lines 57-59. A third process is disclosed in which the hash signature of each elemental data block (h(A), h(B)) is precalculated and stored in a respective array. Each array is then subjected to a second hash function, G, to generate a hash signature (g(h(A)), g(h(B))) of the array of hash signatures. The g(h(A)) signature is communicated and compared to g(h(B)) to isolate the elements of h(A) and h(B) that are different. See col. 7, line 64 - col. 8, line 19.

Examiner has stated that when an out of match condition is determined ("If the signature g(hA) is not identical to g(hB)..."), a second hash is generated. Applicants respectfully submit that this is not an accurate portrayal of the teaching of Livschitz. Livschitz generates a second hash using function G and the G hash is calculated from the first hash signature from function H. However, such a Livschitz second hash is not conditional upon the H function hash signatures being compared and an out of match condition recognized therefrom. Rather, Livschitz teaches that the G function hash proceeds directly from the H function hash signatures being stored in an array and the comparison is only of G function hashes.

Moreover, Livschitz teaches that the first hash is an operation on the data set, itself, (or a subset thereof) and that the second hash is an operation on the first hash. To the contrary, Applicants have claimed that Applicants' first hash is based upon database *values* and Applicants' second hash is based upon database *records*. Also, Applicants have claimed that the "second computational intensity is greater than said first computational intensity and requires a greater amount of communication channel capacity to communicate said second hash than said first

hash". Livschitz does not disclose this feature, either, and in fact acknowledges that the recursive hash communication may consume a large amount of bandwidth.

Examiner has acknowledged that Livschitz does not disclose the "generating the first hash pursuant to a first hash technique of a first computational intensity and based on values, and generating the second hash pursuant to a second hash technique of a second computational intensity, and in which said second computational intensity is greater than said first computational intensity and reqiures a greater amount of communication channel capacity to communicate said second hash than first hash." Yianilos has been introduced to ostensibly teach this missing element.

Yianilos is directed to a database system having the ability to synchronize data within a specified key range between a primary host database and a secondary host database via a limited bandwidth link. See paragraphs [0002], [0005], and [0080]. At commencement of synchronization, a summary is made of all records lying within a given key interval by both databases and a subsequent comparison is made. Upon a mismatch and a large number of records being in the key interval, the key interval is divided into smaller subintervals and summaries of each are sent and compared for each subinterval. See paragraph [0083]. Yianilos describes the hashing function as an associative function, and in particular as an XOR function. See paragraphs [0062] and [0063]. An associative function "takes two p bit strings and produces another p bit string as an output." Paragraph [0064]. Yianilos describes two functions useful for synchronization, a "Get_All_Hashes" function and a "Get_Interval_Hashes" function. The output from the Get_Interval_Hashes function is a list of "triplets" in the form (key_interval,

num_records, hash) over a defined interval having H disjoint subintervals. There is at least one triplet for each subinterval. The hash is an XOR'ed fixed size digest (which digest is stored with each record) of all of the database records in the subintervals. See paragraphs [0070]-[0072]. A Get_Interval_Hashes summary of all the records lying in a key interval in the database is computed and transmitted for comparison. If a mismatch is found, the key interval is partitioned "into smaller sub-intervals and ...summaries for each of the sub-intervals [are sent]. These remote summaries are then compared against corresponding local summaries and the operation is invoked recursively for sub-interval [sic] whose summaries do not match." Paragraph [0083].

Examiner has interpreted Yianilos' teachings as disclosing the first and second hash elements of Applicants' claim, including a "first hash technique of a first computational intensity", "second hash technique of a second computational intensity", "a first hash...based upon the database *values* of the mobile-copy database", "a second hash...based upon the database *records* in the mobile-copy database", and "said second computational intensity is greater than said first computational intensity and requires a greater amount of communication channel capacity to communicate said second hash than said first hash". Applicants respectfully disagree with Examiner's interpretation for the following reasons.

First, Yianilos teaches that the same hash technique is to be used in both the first and the second hashes, that is, Yianilos' Get_Interval_Hashes function is used in both instances. Thus, the requirement that there be a "first" hash technique and a "second" hash technique is not found in Yianilos. Second, Yianilos teaches that the same database information is used in both the first hash and the second hash; Yianilos merely repartitions the database information into a plurality

of smaller subsets of the database. Thus the requirement that the first hash be based upon the database *values* of the database and the requirement that the second hash be based upon the database *records* in the database is not found in Yianilos. Third, Yianilos does not clearly disclose that the second hash requires a greater amount of communication channel capacity to communicate than the first hash. Although Yianilos states: "The synchronization algorithm starts by asking both databases to compute a single summary of all records lying in the key interval. The Get_Interval_Hashes function is invoked for this" (paragraph [0083]), Yianilos defines the Get_Interval_Hashes function as partitioning the interval "into at most H disjoint sub-intervals and returns a list of triplets...The list has one triplet for each subinterval" (paragraph [0071]). Yianilos, therefore, leaves ambiguous the number of sub-intervals are defined in the first hash equivalent and thus leaves ambiguous the amount of channel capacity required.

All of the elements of Applicants' claim 1, therefore, have not been disclosed by Livschitz or Yianilos, taken alone or in combination. For this reason, a *prima facie* case of obviousness has not been made and Applicants believe claim 1 to be allowable over the cited art. For the same reasons, independent claims 15 and 23 are also believed allowable. Dependent claims 2, 9, 12, 13, 18, and 25 are dependent upon presumed allowable independent claims and are themselves presumed allowable for this reason.

Dependent claims 21, 22, and 24 have been rejected under 35 U.S.C.§103(a) as being unpatentable over Livschitz and Yianilos and further in view of U.S. Patent Application Publication No. 2002/0120648 by Ball ("Ball"). Examiner observes that the combination of Livschitz and Yianilos do not disclose that the first hash technique comprises a checksum. Ball

is directed to the presentation of a current version of a document to a user with an indication of modifications made to the document since the last viewing of the document. In looking for changes, a preliminary check is made of the dates of modification and checksums of all of the pages of the document, where a checksum is the numerical sum of all of the characters in a line or on a page. See paragraphs [0049]-[0051].

Applicants assert, however, that the whole of the invention has not been considered in incorporating Ball. When the whole of at least the first hash element is examined with the limitations of both claim 1 and claim 21 incorporated, one finds: "generate a first hash pursuant to a first hash technique of a first computational intensity *comprising a checksum process* and based upon the database values of the mobile-copy database...whereby an out of match condition between the mobile-copy database values and the network-copy database values is determined". As explained above, Ball describes using checksums to locate a change on a page or a particular line of a page. Ball does not disclose the use of a checksum to determine whether the entirety of a document has been modified. Accordingly, the whole of Applicants' invention has not been disclosed in the combination of Livschitz, Yianilos, and Ball, and claims 21, 22, and 24 (with similar limitations found in each), along with their independent base claim limitations, are believed to be allowable.

Examiner has rejected claims 5-7 under 35 U.S.C.§103(a) as being unpatentable over Livschitz and Yianilos and further in view of U.S. Patent No. 5,809,494 to Nguyen. Examiner has also rejected claims 14, 19, and 20 under 35 U.S.C.§103(a) as being unpatentable over Livschitz and Yianilos and further in view of U.S. Patent No. 5,684,990 to Boothby. Claims 5-7,

14, 19, and 20 are dependent upon presumed allowable independent claims and, as such, are themselves presumed allowable.

In light of the foregoing remarks, Applicants believe all of the pending claims to be allowable. Examiner is respectfully urged to withdraw the claims rejection, reconsider the present Application, and pass the present Application to allowance.

Respectfully submitted,

/ Robert H. Kelly /

Robert H. Kelly
Registration No. 33,922

KELLY & KRAUSE, L. P.
6600 LBJ Freeway, Suite 275
Dallas, Texas 75240
Telephone: (214) 446-6684
Fax: (214) 446-6692